MeTime

www.nischaalc.github.io/MeTime/

Cameron Spencer, Roger White, Nischaal Cooray
CS 275, Summer 2015 – Final Project

Project Description

MeTime helps you plan out your week based on events in your calendar, automatically generating study and practice times for Classes and Goals you add to help you keep your grades up, while still keeping your free time in mind. MeTime has three components, an Android app, a Web app and a Web Server. Although we do not use a persistence server, data for the app is drawn directly from Google Calendar and any updates by the user are added to Google Calendar.

Project Features

Add Events, Goals and Classes to your calendar through the MeTime Web app or Android app. When a Goal or Class is added, MeTime will automatically generate times for studying or practicing for that Goal/Class based on the priority that you assign to it. The higher the priority, the more study/practice sessions will be generated and added to your calendar.

Usage

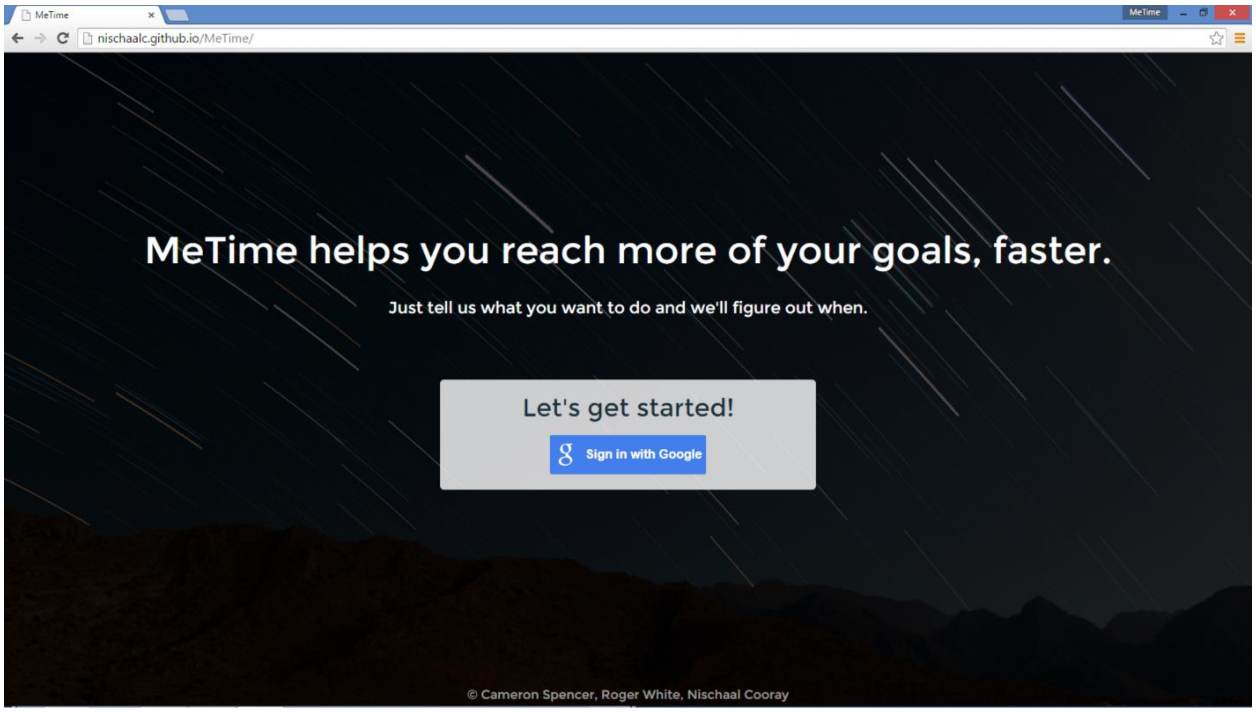
MeTime is simple to use. Revolving around three core items, all you have to do is add one of the three and MeTime does the rest. When adding a class, simply enter the name, start time, end time, priority and the days of the week it recurs and the class will be added to your calendar for those days at the specified time. The same applies for Events. When adding a Goal, you only need to specify the name and priority of the Goal and MeTime will find free times in your calendar for items to be added to. Similarly, when a Class is added, MeTime determines the best study times for you based on the priority that you assigned to the class, adding these study times to your calendar in order to best fit your schedule.

MeTime displays events for the next 7 days in both the web and Android apps, giving you a good look at what is coming up for your day and week and giving you the opportunity to plan around any study sessions and goal practice that may be coming up.

Design and Implementation

Our design principle for MeTime was relatively straightforward and simple. We wanted an app that was user friendly, while also being informative and useful, with the user having to do as little work as possible.

For the web app, this was accomplished through the use of various libraries that made it easy for us to display information to the user in a condensed, concise form, while allowing the user to interact with the on screen components in order to get more information if required. We used jQuery, Toastr and FullCalendar in order to provide this experience to the user.

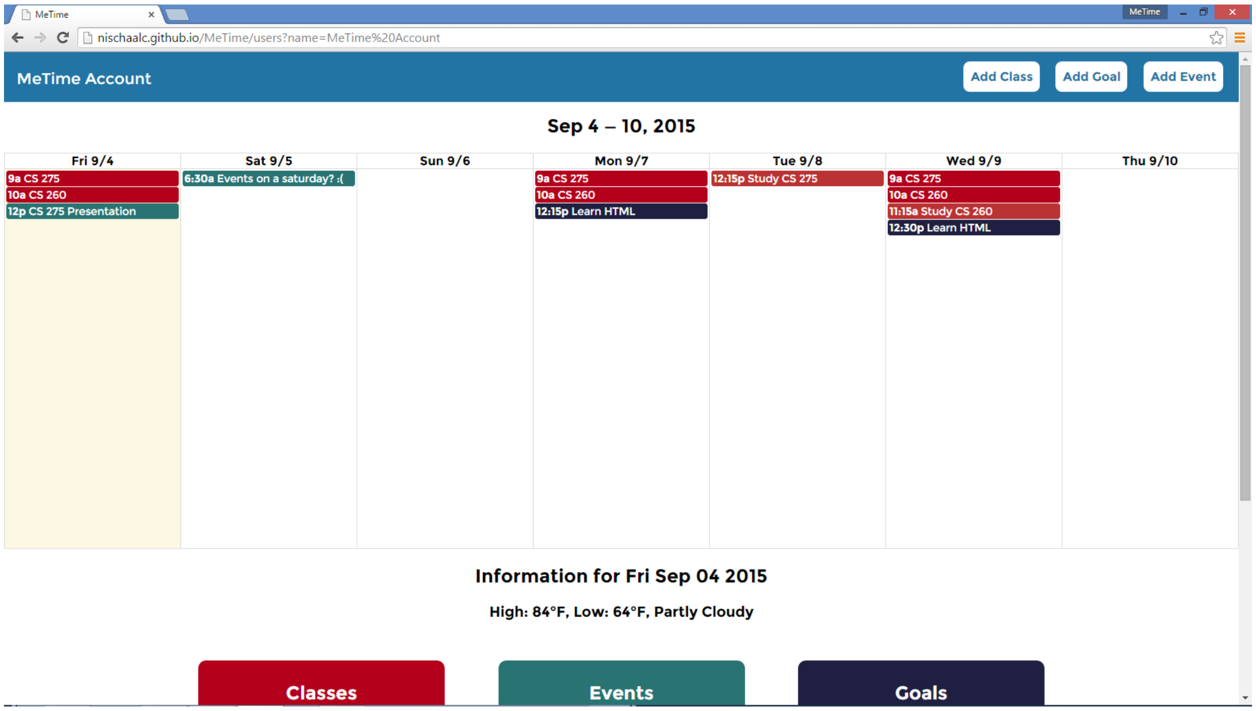


MeTime helps you reach more of your goals, faster.

Just tell us what you want to do and we'll figure out when.

Let's get started!

© Cameron Spencer, Roger White, Nischaal Cooray



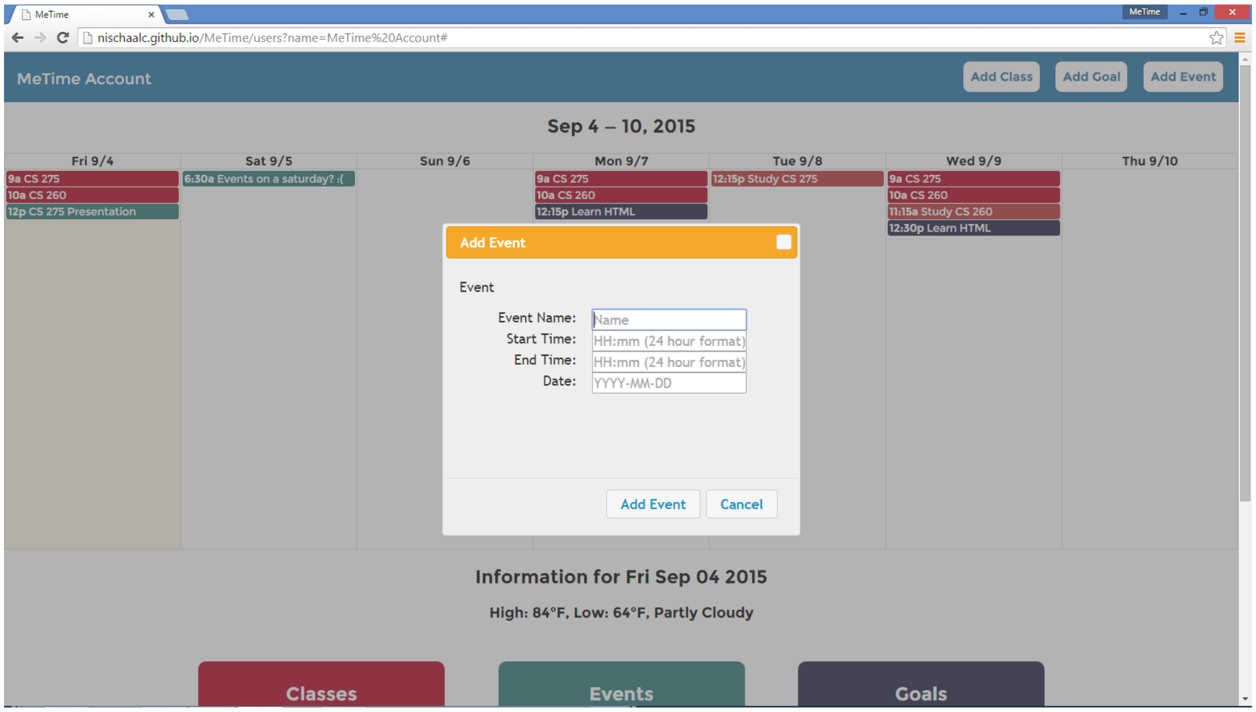
MeTime Account

Sep 4 – 10, 2015

Fri 9/4	Sat 9/5	Sun 9/6	Mon 9/7	Tue 9/8	Wed 9/9	Thu 9/10
9a CS 275	6:30a Events on a saturday? :(		9a CS 275	12:15p Study CS 275	9a CS 275	
10a CS 260			10a CS 260		10a CS 260	
12p CS 275 Presentation			12:15p Learn HTML		11:15a Study CS 260	
					12:30p Learn HTML	

Information for Fri Sep 04 2015

High: 84°F, Low: 64°F, Partly Cloudy


MeTime

nischaalc.github.io/MeTime/users?name=MeTime%20Account#

MeTime Account

Add Class | Add Goal | Add Event

Sep 4 – 10, 2015

Fri 9/4: 9a CS 275, 10a CS 260, 12p CS 275 Presentation
Sat 9/5: 6:30a Events on a saturday? :(
Sun 9/6
Mon 9/7: 9a CS 275, 10a CS 260, 12:15p Learn HTML
Tue 9/8: 12:15p Study CS 275
Wed 9/9: 9a CS 275, 10a CS 260, 11:15a Study CS 260, 12:30p Learn HTML
Thu 9/10

Add Event

Event

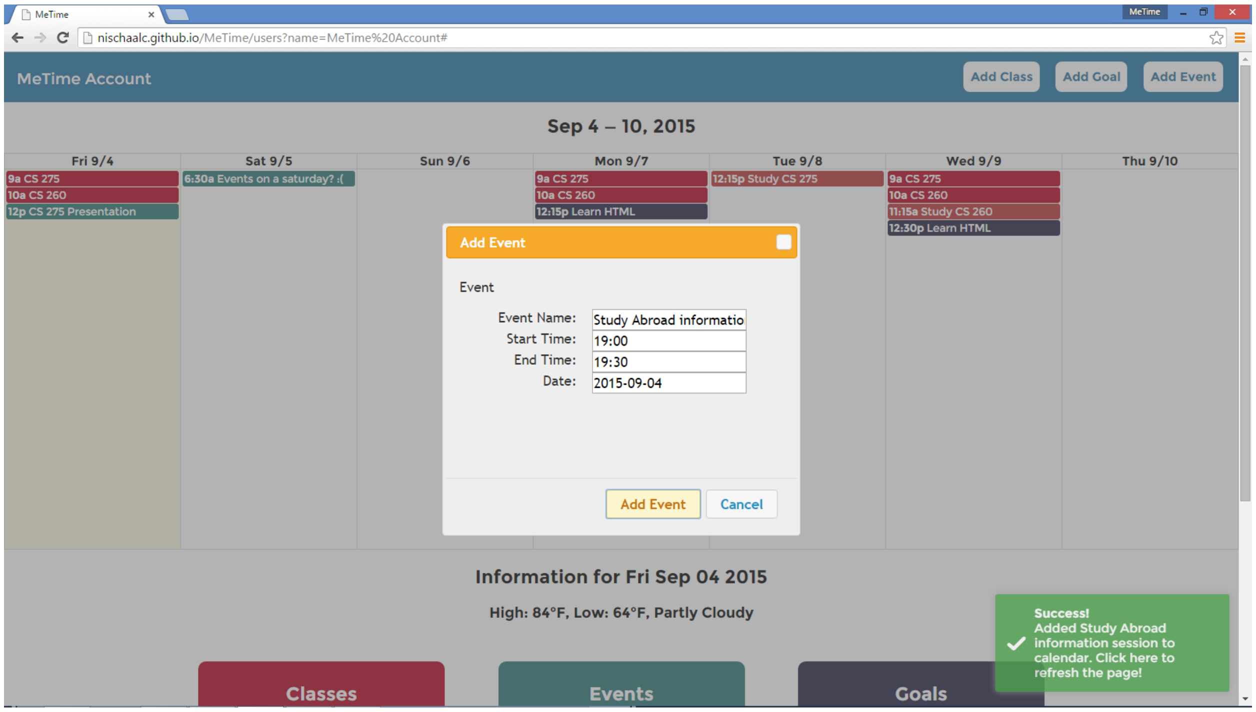
Event Name: Name
Start Time: HH:mm (24 hour format)
End Time: HH:mm (24 hour format)
Date: YYYY-MM-DD

Add Event | Cancel

Information for Fri Sep 04 2015

High: 84°F, Low: 64°F, Partly Cloudy

Classes | Events | Goals


MeTime

nischaalc.github.io/MeTime/users?name=MeTime%20Account#

MeTime Account

Add Class | Add Goal | Add Event

Sep 4 – 10, 2015

Fri 9/4: 9a CS 275, 10a CS 260, 12p CS 275 Presentation
Sat 9/5: 6:30a Events on a saturday? :(
Sun 9/6
Mon 9/7: 9a CS 275, 10a CS 260, 12:15p Learn HTML
Tue 9/8: 12:15p Study CS 275
Wed 9/9: 9a CS 275, 10a CS 260, 11:15a Study CS 260, 12:30p Learn HTML
Thu 9/10

Add Event

Event

Event Name: Study Abroad informatio
Start Time: 19:00
End Time: 19:30
Date: 2015-09-04

Add Event | Cancel

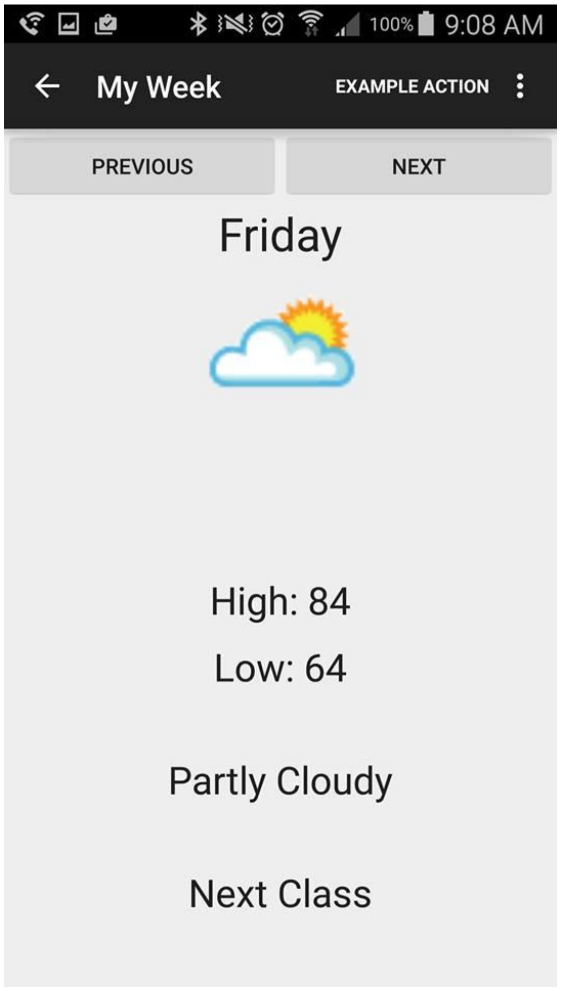
Information for Fri Sep 04 2015

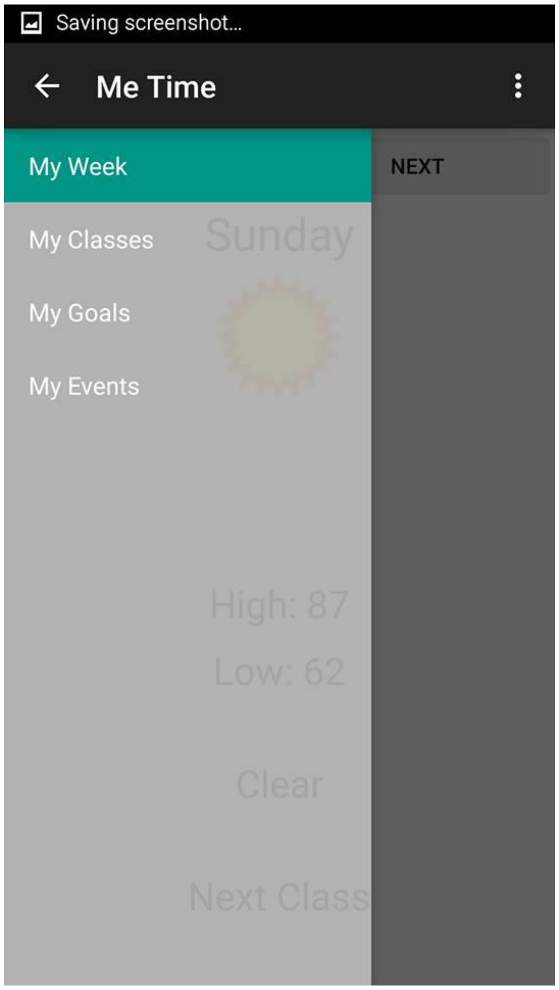
High: 84°F, Low: 64°F, Partly Cloudy

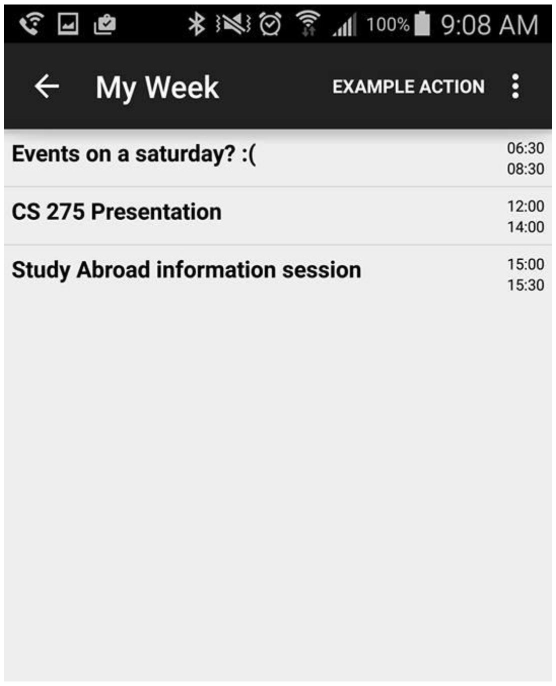
Classes | Events | Goals

Success!
Added Study Abroad information session to calendar. Click here to refresh the page!

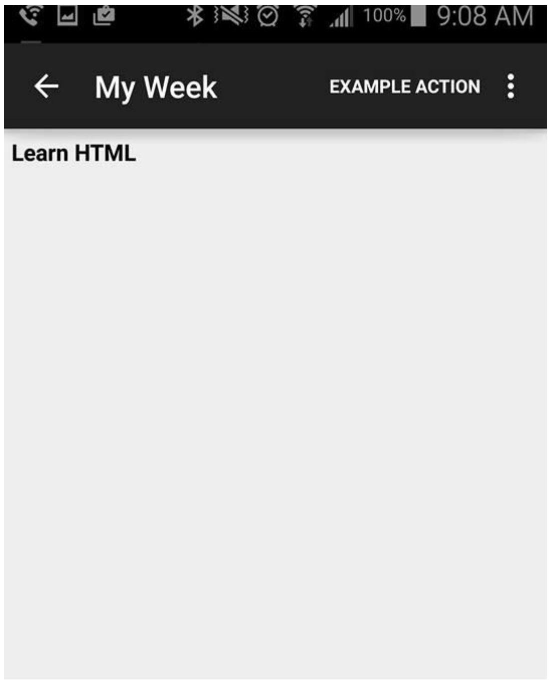
For the Android app, our design remained simple. The user is presented with a few different views to interact with and view items on their calendar.







Events on a saturday? :(	06:30 08:30
CS 275 Presentation	12:00 14:00
Study Abroad information session	15:00 15:30



Learn HTML

Implementation of the web app was relatively simple since we all had experience with web development before and the use of the various external libraries listed above removed the necessity to implement certain features as they were already provided by said libraries.

Implementing the server was a more difficult task as none of us had any experience with running a server. We chose to go with Python and as our server side language, due to its power, simplicity and numerous libraries that we could use to enhance the features that we were providing to the user. Both the web and android apps rely on the server to do all the processing required by each component. As a result, the only processing happening in the web and android apps are to allow the user to view their calendar items and to add items to the calendar as they desire.

The server is powered by Python and runs a Flask instance on Heroku. We chose Heroku due to it's ease of use, since it runs code that you commit to a Git repository and Flask was the best available option to run a Python server. The server does all the processing required by the apps, ranging from determining which dates and times to add study/practice sessions to returning a list of all events on the users calendar.

Development Log

- **Issue –** Server was unable to interpret POST requests due to CORS issue.

  Fix – Implemented Flask CORS fix found on the Flask website.

- **Issue –** Android app unable to make POST requests to the server. Server returns 500 error, however the web app works as expected for the same data.

  Fix – As of September 4th, no fix has been found. We are unsure if this is a server related issue or Android related issue.

- **Issue –** Times that items are added sometimes do not take time zone into account and will be added to the calendar as UTC times, which causes the web app to display them 4 hours earlier than they are meant to be. The android app displays the same events at the correct times.

  Fix – Most likely an issue with the web app JavaScript for parsing the DateTime object returned by the server. No fix has been implemented yet.

- **Issue –** Parsing POST request sent to the server.

  Fix – Had to look up Flask REST guidelines and various Stack Overflow posts in order to get the POST endpoint working as expected.

- **Issue –** Making sure a study/practice event wasn't added at the same time as an existing item.

  Fix – Used Python's datetime library to compare the dates and times for each item before adding them to the calendar.

- **Issue –** Our day of the week viewer always displayed Saturday as the current day

  Fix – Call to Calendar's DAY_OF_WEEK returned 7 every time it was called (messing up our day of the week view). The initialization was in the wrong place

- **Issue –** The animations for the day of the week viewer, by default would only animated in from the left and out to the right (made it awkward when using previous and having the animation appear to be going to the next element)

  Fix – Created our own XML files in animation that would animate in from the right and out from to the right

- **Issue –** In each day view when High and Low temperature TextViews were placed side by side, they would not be updated after the information from Wunderground was received

  Fix – I was unable to get the TextViews to work side by side so I placed them one over the other and it worked

- **Issue –** Sending a POST request from the app to the server

  Fix – ***unable to fix* -** The server would successfully receive the request, but would then through a 500 error, meaning the server encountered an unexpected condition that prevented it from executing the request – after comparing the request from the website (Website POST is working), there did not seem to be any difference. More time would be needed to debug exactly what is going wrong here.

- **Issue** – Difficulty running Async Tasks inside Activity Fragments.

  Fix – Using more activities and running async tasks in Main Activities.

- **Issue –** Gesture Listener for swipe navigation interfered with onClick listener.

  Fix – Decided to use button navigation.